FAST VERY LARGE VOCABULARY RECOGNITION BASED ON COMPACT DAWG-STRUCTURED LANGUAGE MODELS

K. Georgila, K. Sgarbas, N. Fakotakis, G. Kokkinakis

Wire Communications Laboratory
Electrical and Computer Engineering Dept.
University of Patras, Greece
{rgeorgil, sgarbas, fakotaki, gkokkin}@wcl.ee.upatras.gr

ABSTRACT

In this paper we present a method for building compact lattices for very large vocabularies, which has been applied to surname recognition in an Interactive telephone-based Directory Assistance Services system[1]. The method involves the construction of a non-deterministic DAWG, which is eventually transformed into a phoneme lattice in Entropic's HTK Application Programming Interface (HAPI) format. Incremental construction functions are used for the creation and update of the DAWG, whereas an algorithm for converting the DAWG into the HAPI format is presented. Furthermore, trees, graphs and full-forms (whole words with no merging of nodes) are compared in a straightforward way under the same conditions, using the same decoder (HAPI MVX) and the same vocabularies. Experimental results showed that as we go from full-form lexicons to trees and then to graphs the size of the recognition network is reduced and therefore the recognition time too. However, recognition accuracy is retained since the same phoneme combinations are involved.

1. INTRODUCTION

In applications such as automatic directory information services, a speech recognizer is expected to be able to handle very large vocabularies. Moreover, in most applications these vocabularies are expected to be "open-set lexicons" meaning that more words, e.g. surnames, first names, city names etc., may need to be added later. Therefore, an efficient method able to cope with the above constraints should provide a real-time search operation over the whole vocabulary structure and a means of easy vocabulary augmentation.

Most speech recognition systems that have to deal with very large vocabularies, use a tree structure (i.e. trie) for the word lexicon in order to restrict the search space. By sharing phones across different words (as opposed to using a separate instance for every phone in the pronunciation of each word), the lexical tree provides a compact representation of the acoustic-phonetic search space, as well as a mechanism to efficiently handle multiple pronunciations of the same word. However, trees are not the optimal way to represent lexicons, due to their inadequacy to exploit common word suffixes. For this reason, the use of DAWG (Directed Acyclic Word Graph) structures is more appropriate. A DAWG is a special case of a finite-state automaton where no loops (cycles) are allowed. It has a "start" and an "end" node and all possible paths between them form the words of the lexicon. DAWG structures have been successfully used for storing large vocabularies in speech recognition. Hanazawa et al. [1] used an incremental method [2] to generate deterministic DAWGs. The aforementioned method was applied on a 4000-word vocabulary in a telephone directory assistance system. However, in [1] the comparison between the tree and the DAWG was made using different decoding algorithms. Thus the efficiency of the DAWG was not shown under the same conditions.

In this paper we also use incremental construction of DAWGs in order to be able to update them without having to build them from scratch in every change. We have used the incremental construction algorithms described in [3] because we are particularly interested in non-deterministic DAWGs since they appear to be even more compact than the minimal deterministic ones [4],[3]. After the DAWG is constructed it is converted to a lattice in the format that the decoder expects, where the labels are on the nodes instead of the arcs. In addition, we compare the use of trees, graphs and full-forms (whole words with no merging of nodes) in a straightforward way under the same conditions, using the same decoder (Entropic's HAPI MVX) and the same vocabularies. In our application, the EU Project IDAS, the recognizer must distinguish between 257,198 distinct surnames that correspond to 4,597,382 entries in the directory of the Greek Organization of Telecommunications. By restricting the search space to the most frequent 100,000 ones, 93.66% of the directory's listings is covered. The use of DAWGs proves to reduce recognition time significantly without affecting accuracy. The efficiency of the algorithm becomes more evident as the size of the vocabulary increases. The idea is general and works in any case where the target is to find the best hypothesis among a great number of words.

The paper is organized as follows: The lattice construction method is explained in Section 2. Section 3 describes how the decoder processes the word and phoneme lattices and explains why recognition accuracy is retained while time decreases. Results of performed experiments are given in Section 4. Finally, a short summary and some conclusions are given in Section 5.

2. LATTICE CONSTRUCTION

A word lattice consisting of surnames is replaced by a phoneme lattice that can produce the phonetic transcriptions of all the above surnames (Figures 2a, 2d). Thus, a lexicon of surnames in phonetic transcription (Figure 2a), is first transformed into a

[1] The EU Project LE-48315, IDAS.

Directed Acyclic Word Graph (DAWG) (Figure 2c). There are several types of equivalent DAWGs that can be produced from a set of strings using different algorithms. Since in this stage we wish to have the most compact structure, we choose to build a non-deterministic DAWG using the incremental algorithm described in [3] because the resulting DAWG proves to be even more compact than the corresponding minimal deterministic one [4],[3]. In this way new words can be inserted in the DAWG without building and minimizing it from scratch.

```
create a starting lattice-node P[start]
for every DAWG-link L[i] with label A {
        create a new lattice-node P[i]
        assign label A to P[i]
}
create an ending lattice-node P[end]

let N[start] = the starting DAWG-node
for every DAWG-link L[i] leaving N[start] {
        create a new lattice-link T[i] from P[start] to lattice-node P[i]
}
let N[end] = the ending DAWG-node
for every DAWG-link L[i] entering N[end] {
        create a new lattice-link T[i] from lattice-node P[i] to P[end]
}

for every DAWG-node N[k] except the N[start] and the N[end] {
        for every DAWG-link L[i] entering N[k] {
                for every DAWG-link L[j] leaving N[k] {
                        create a new lattice-link T[r] from lattice-node P[i] to P[j]
                } } }
```

Figure 1: A straightforward procedure that transforms the DAWG into the HAPI lattice format.

Our algorithm produces the DAWG of Figure 2c, where simple monophone pronunciations label the transitions between nodes. The next stage of the method is to convert these structures into the format that is accepted by the HAPI decoder (see Section 3), where the labels are on the nodes (Figure 2d). This conversion is not trivial, i.e. we cannot just convert the transitions to nodes and vice-versa retaining the topology of the lattice, considering that some transitions with similar labels may

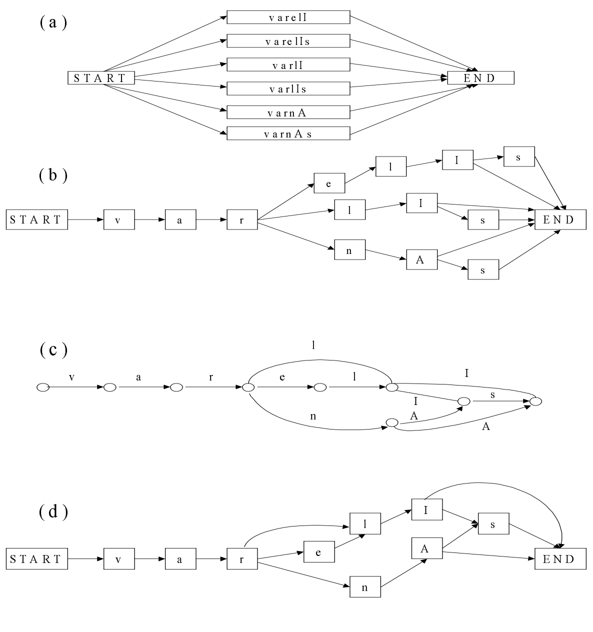

Figure 2: (a) Word lattice, (b) HAPI's tree-based lattice, (c) DAWG produced by our incremental algorithm, (d) DAWG-based lattice produced by our straightforward procedure.

enter or leave the same nodes. For the conversion we have used the procedure of Figure 1, which (although it does not result to the most compact representation) produces correct lattices giving impressive results as shown in Section 4.

Figure 2 shows how the algorithm works for 6 surnames. The first network is a word lattice in the HAPI format, where each node corresponds to a surname. The second network is the tree produced by HAPI and the third network is the DAWG created by our incremental algorithm while the fourth one is the transformed DAWG into the HAPI format. If the surnames in Figure 2 had multiple pronunciations, they would be treated as different words by the algorithm. Using the above lattice reduction method, we get an equivalent but more compact lattice, which results to faster search. In both the tree-based and DAWG-based lattices several words have a common path, thus recognition is substantially accelerated in comparison to the word lattice, when the same recognizer is used in all lattices. In addition the DAWG-based lattice is more compact than the tree-based one since common suffixes are also merged.

3. DECODING PROCESS

The speech recognizer we used is Entropic's HTK Application Programming Interface (HAPI). All the possible speaker utterances form a network defined using the HAPI Standard Lattice Format (SLF). An SLF network is a text file, which contains a list of nodes representing words or sub-words or even single letters and a list of arcs representing the transitions between nodes. Given the set of the acoustic models (HMMs), the SLF file and the corresponding dictionary, which contains the phonetic transcriptions of the words, sub-words or letters included in SLF, HAPI produces the N-best hypotheses.

In all our experiments we have used three different types of SLF networks together with their corresponding dictionaries. In the first case the nodes are full surnames. E.g. in Figure 2a the word lattice consists of 6 words. The corresponding dictionary contains the monophone transcriptions of these words. During decoding the HAPI expands the word lattice of Figure 2a to the network of Figure 3. Each word in the word lattice is transformed into a word-end node preceded by the sequence of model nodes corresponding to the word's pronunciation as defined in the above dictionary. Monophones are expanded to context-dependent triphones and there is also cross-word context expansion between the sil model of the START and END nodes and the models that form the full surnames. The second type of SLF file gives the structure of the tree-based phoneme network depicted in Figure 2b and the third one describes the DAWG-based phoneme network shown in Figure 2d. Now the dictionary consists of START and END corresponding to sil and monophones having the same pronunciation. The network of Figure 2b is expanded to the one in Figure 4, and the network of Figure 2d to the one in Figure 5. If we compare the networks of Figures 3, 4 and 5 it is clear that the second and third networks are more compact and contain fewer model nodes than the first one. However, the number of word-end nodes increases since each monophone is considered as a distinct word. The conducted experiments described in Section 4 prove that as the size of the vocabulary increases the

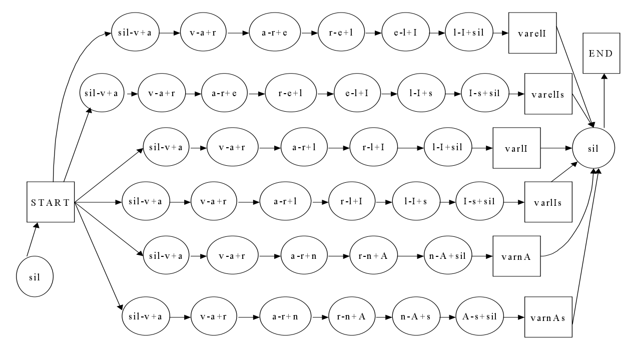
sil-v+a v-a+r a-r+e r-e+l e-l+I l-I+sil varelI
sil-v+a v-a+r a-r+e r-e+l e-l+I l-I+s I-s+sil varelIs
sil-v+a v-a+r a-r+l r-l+I l-I+sil varlI
sil-v+a v-a+r a-r+l r-l+I l-I+s I-s+sil varlIs
sil-v+a v-a+r a-r+n r-n+A n-A+sil varnA
sil-v+a v-a+r a-r+n r-n+A n-A+s A-s+sil varnAs
START
sil
sil
END

Figure 3: The expansion of the word lattice to triphone model nodes and to word-end nodes.

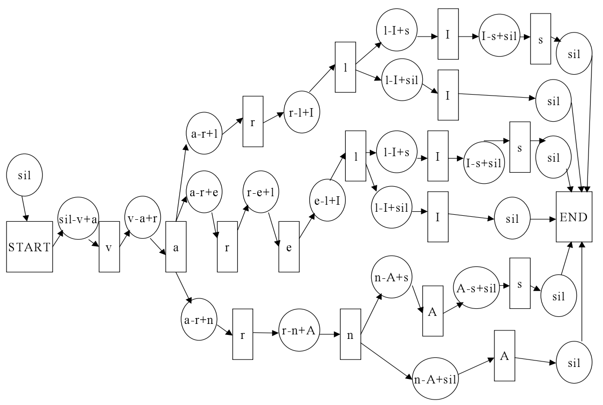

Figure 4: The expansion of the tree-based phoneme lattice to triphone model nodes and word-end nodes (monophones).

total number of nodes and links of an expanded phoneme lattice (tree-based or DAWG-based) is getting smaller than the one of an expanded word lattice. This is something expected because in both cases, links are merged in order to produce an efficient network. Therefore, as word lattices grow larger we will reach a point where their equivalent phoneme lattices have fewer word-end nodes due to the merging process. In addition, if some context-dependent triphones have been tied during training, their model nodes are merged. This could lead to further decrease of the phoneme lattice's size. It should be noticed that sometimes during the expansion NULL nodes must be inserted. But even if they increase the number of nodes and links in some cases they do not add to the processing time as explained in the following. DAWG-based lattices are more compact than the corresponding tree-based ones since not only prefixes are merged as in trees, but also suffixes.

The Viterbi beam search algorithm traverses the expanded network and estimates the acoustic probabilities until it reaches a word-end node. At this point, it combines the above probabilities with the language modelling probability of the word in the word-end node. In our case we don't use transition probabilities between words. Thus, the final scores depend only on the acoustic probabilities and since both the word and phoneme networks give the same sequences of models in each

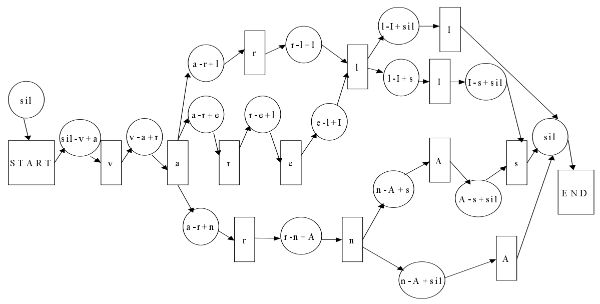

Figure 5: The expansion of the DAWG-based phoneme lattice to triphone model nodes and to word-end nodes (monophones).

path, the recognition accuracy is not affected. Although a phoneme lattice (tree or DAWG) becomes smaller than a word lattice only after surnames exceed a certain number, the recognition time is improved for all vocabulary sizes. This is explained, if we take into account the fact that the only reason the expanded phoneme lattice could have more nodes and links than the corresponding word one, is the additional number of word-end or NULL nodes. However, the computational cost at a word-end or NULL node is very small compared to the cost at a triphone model node even if we use transition probabilities between words, which is not our case.

4. PERFORMED TESTS

In order to test the efficiency of our proposed method we used 106 different surnames spoken by 4 speakers (2 male and 2 female). We performed three types of experiments. In the first type we used a word network like the one depicted in Figure 3, in the second a tree-based phoneme network (Figure 4) and in the third one a DAWG-based phoneme network as the one in Figure 5. Tests also differed in the number of words contained in the dictionary, that is in the size of the word and phoneme networks. The performed experiments had three goals. Firstly to examine how the number of nodes and links changes according to the vocabulary size and prove that after a certain point it decreases for tree-based and furthermore for DAWG-based lattices. Secondly, to show in practice that recognition accuracy is retained and thirdly to prove that processing time decreases for all vocabulary sizes (for trees and furthermore for DAWGs).

Figure 6 shows the number of nodes and links for 9 different vocabularies described by 2 numbers. The first one is the number of distinct pronunciations and the second is the number of distinct surnames. The first number is always greater than or equal to the second one since some words are pronounced in multiple ways. In Figures 7 and 8 we can see the recognition time gain in seconds between word networks and trees, and trees and DAWGs, respectively. The time gain is shown for 6 different pruning levels and the 9 different vocabulary sizes mentioned above. L0 means that there is no pruning and the search is exhaustive. As we go from L1 to L5 pruning increases and more paths are abandoned before their full search. We have used two pruning thresholds one for model nodes and the other for word-end nodes. We have chosen the two thresholds to be equal. The results of the tests confirm that the accuracy is

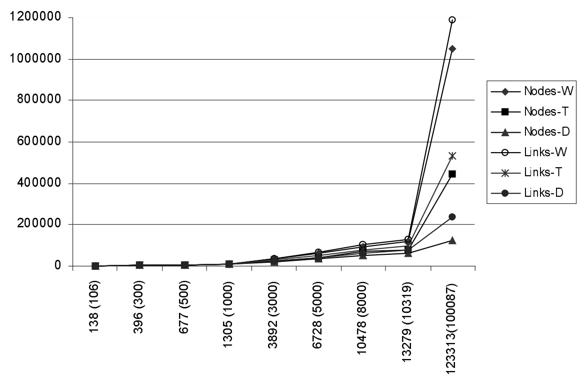

Figure 6: The number of nodes and links (y-axis) for the word (W), tree-based (T) and DAWG-based(D) phoneme lattices in connection with the vocabulary size.

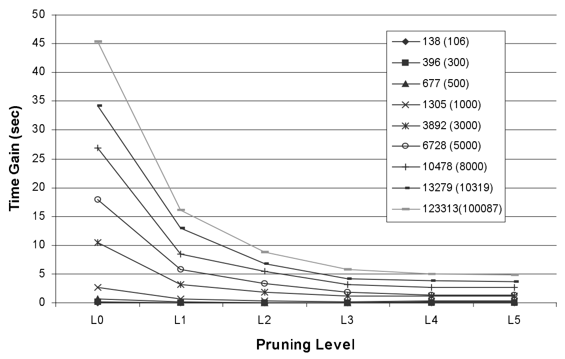

Figure 7: The recognition time gain in seconds between word lattices and trees (6 pruning levels, 9 vocabulary sizes).

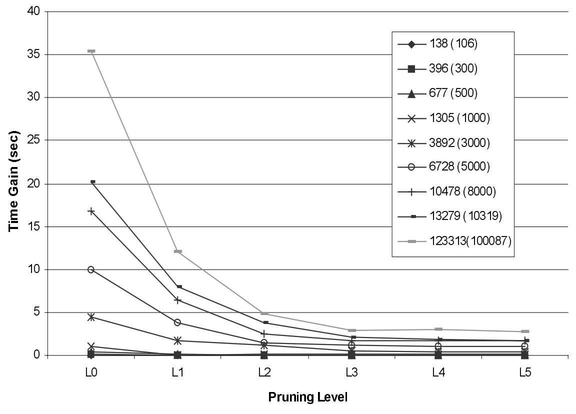

Figure 8: The recognition time gain in seconds between trees and DAWGs (6 pruning levels, 9 vocabulary sizes).

retained in all cases ranging from 64 correct recognitions for high pruning and large vocabularies to 103 for small or no pruning and small vocabularies. Pruning level L3 gives the best trade-off between accuracy and processing time. According to the experiments, when trees or DAWGs are used, if we select a word-end pruning threshold higher than the model one, accuracy drops (maximum 8% for large vocabularies). This is justified by the fact that each word is equivalent to a model, thus a greater pruning threshold for word-end nodes entails an increased pruning threshold for model nodes even if the model pruning threshold is smaller. It should also be noticed that experiments showed that if the word-end pruning threshold in word lattices is greater than the word-end pruning in phoneme lattices, while both lattice types have the same model pruning threshold, we get the same accuracy. In this case one would expect that since pruning increases for word lattices, time would decrease. This is true but it still remains significantly greater than the processing time of phoneme lattices that have smaller pruning thresholds. Thus even if we need smaller pruning thresholds in phoneme lattices to get the same accuracy we have in word lattices with higher pruning thresholds, recognition time in phoneme lattices (tree-based or DAWG-based) is still significantly smaller. The above observation justifies the efficiency of using trees and furthermore DAWGs.

5. SUMMARY AND CONCLUSIONS

In this paper we presented an algorithm for reducing the size of word lattices by replacing them by phoneme ones, in such a way that all the possible paths of the phoneme lattices produce the phonetic transcriptions of the words in the word lattices. Our algorithm produces DAWG-based networks. In order to test the efficiency of our method we have conducted experiments and have compared networks based on full-formed words, trees and DAWGs under the same conditions (using the same decoder and vocabulary). We have proved that the size of trees and DAWGs is reduced after a certain point compared to the word lattices and that recognition accuracy is retained while processing time decreases significantly for all vocabulary sizes. It has also been shown that DAWG-based networks are more compact than tree-based ones and lead to smaller recognition times. In our experiments we have used context-dependent triphones and lattices containing only surnames, but the algorithm could work efficiently for other models too, such as syllables, and for lattices that consist of surnames with their context.

6. REFERENCES

1. Hanazawa K., Minami Y., and Furui S., “An efficient search method for large-vocabulary continuous-speech recognition”, *ICASSP ’97*, pp. 1787-1790, Munich, Germany.

2. Aoe J., Morimoto K., and Hase M., “An algorithm of compressing common suffixes for trie structures”, *Trans. IEICE*, Vol. J75-D-II No. 4, April 1992, pp. 770-799.

3. Sgarbas K., Fakotakis N., and Kokkinakis G., “Two algorithms for incremental construction of directed acyclic word graphs”, *International Journal on Artificial Intelligence Tools,* Vol. 4, No. 3, pp. 369-381, 1995.

4. Sgarbas K., Fakotakis N., and Kokkinakis G., “Optimal Insertion in Deterministic DAWGs”, *Submitted to Theoretical Computer Science*, Elsevier.